Horse_or_Human

June 21, 2020

[1]:

```
import os
import zipfile
```

[2]:

```
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras
```

[3]:

```
local_zip = 'F:/Courses/COURSERA/Tensorflow in practice/Course1/Week3/
 ↪horse-or-human.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('/Week3/horse-or-human')
zip_ref.close()
```

[4]:

```
# Direactory with training horse pictures
train_horse_dir = os.path.join('/Week3/horse-or-human/horses')
# Directory with training human pictures
train_human_dir = os.path.join('/Week3/horse-or-human/humans')
```

[5]:

```
train_horse_names = os.listdir(train_horse_dir)
print(train_horse_names[:10])

train_human_names = os.listdir(train_human_dir)
print(train_human_names[:10])
```

```
['horse01-0.png', 'horse01-1.png', 'horse01-2.png', 'horse01-3.png',
'horse01-4.png', 'horse01-5.png', 'horse01-6.png', 'horse01-7.png',
'horse01-8.png', 'horse01-9.png']
['human01-00.png', 'human01-01.png', 'human01-02.png', 'human01-03.png',
'human01-04.png', 'human01-05.png', 'human01-06.png', 'human01-07.png',
'human01-08.png', 'human01-09.png']
```

[6]:

```
print('total training horse images:', len(os.listdir(train_horse_dir)))
print('total training human images:', len(os.listdir(train_human_dir)))
```

```
total training horse images: 500
total training human images: 527
```

[7]:

```
%matplotlib inline

import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Parameters for our graph; we'll output images in a 4x4 configuration
nrows = 4
ncols = 4

# Index for iterating over images
pic_index = 0
```

[8]:

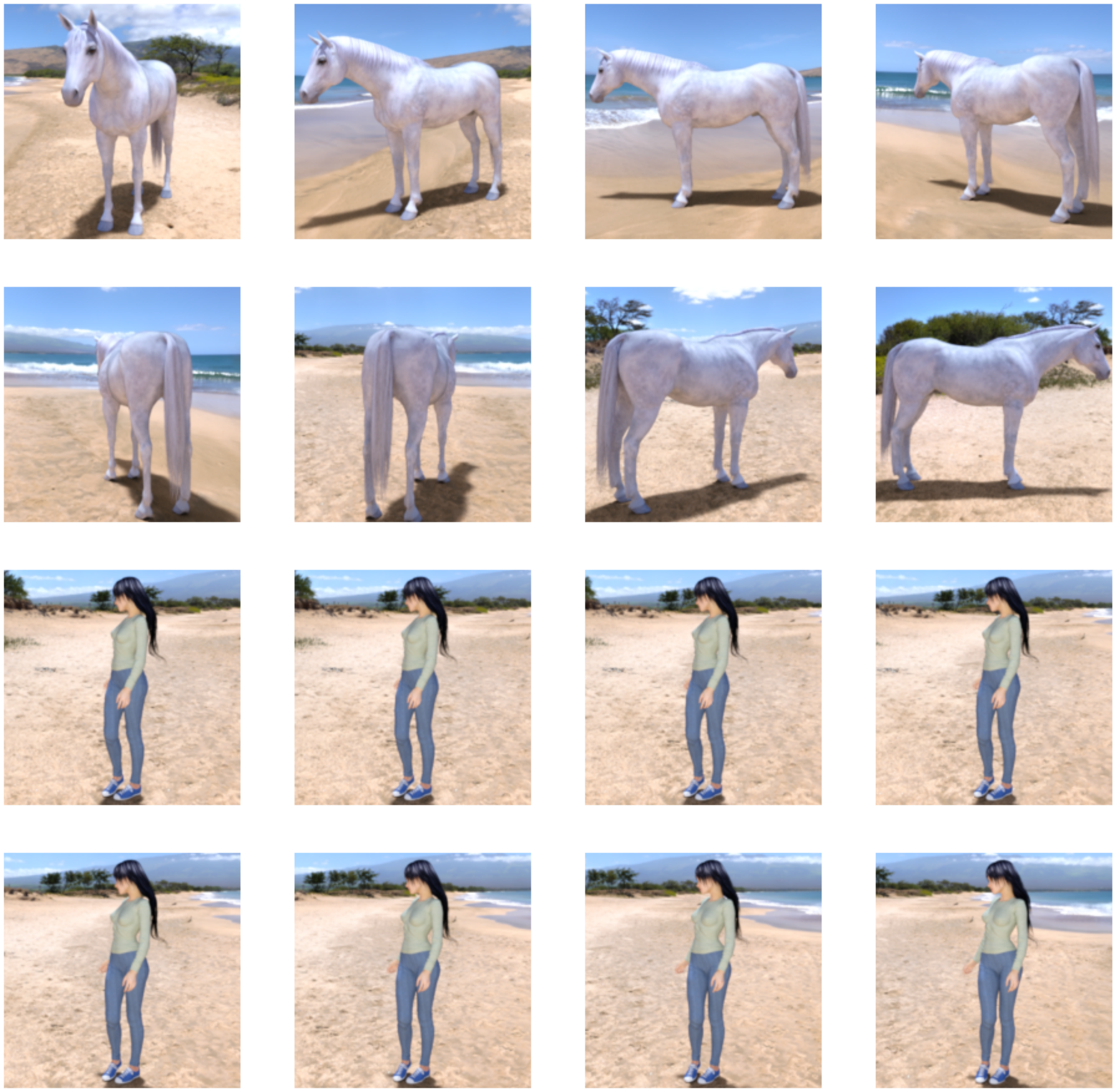
```
# Set up matplotlib fig, and size it to fit 4x4 pics
fig = plt.gcf()
fig.set_size_inches(ncols * 4, nrows * 4)

pic_index += 8
next_horse_pix = [os.path.join(train_horse_dir, fname)
                for fname in train_horse_names[pic_index-8:pic_index]]
next_human_pix = [os.path.join(train_human_dir, fname)
                for fname in train_human_names[pic_index-8:pic_index]]

for i, img_path in enumerate(next_horse_pix+next_human_pix):
  # Set up subplot; subplot indices start at 1
  sp = plt.subplot(nrows, ncols, i + 1)
  sp.axis('Off') # Don't show axes (or gridlines)

  img = mpimg.imread(img_path)
  plt.imshow(img)

plt.show()
```



[22]:

```
from keras import layers
from keras.models import load_model
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from PIL import Image
from keras.preprocessing import image
```

[10]:

```
model = keras.Sequential([
    tf.keras.layers.Conv2D(16, (3, 3), activation = 'relu', input_shape = (300, 300, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3, 3), activation = 'relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation = 'relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation = 'relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation = 'relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation = 'relu'),
    tf.keras.layers.Dense(1, activation = 'sigmoid')
])
```

[11]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 298, 298, 16)      448

_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 149, 149, 16)      0

_________________________________________________________________
conv2d_1 (Conv2D)            (None, 147, 147, 32)      4640

_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 73, 73, 32)        0

_________________________________________________________________
conv2d_2 (Conv2D)            (None, 71, 71, 64)        18496

_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 35, 35, 64)        0

_________________________________________________________________
conv2d_3 (Conv2D)            (None, 33, 33, 64)        36928

_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 16, 16, 64)        0

_________________________________________________________________
conv2d_4 (Conv2D)            (None, 14, 14, 64)        36928

_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 64)          0

_________________________________________________________________
flatten (Flatten)            (None, 3136)              0

_________________________________________________________________
dense (Dense)                (None, 512)               1606144

_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 1,704,097
Trainable params: 1,704,097
Non-trainable params: 0

_________________________________________________________________
```

[12]:

```
model.compile(loss='binary_crossentropy',
              optimizer=RMSprop(lr=0.001),
              metrics=['accuracy'])
```

[13]:

```
train_datagen = ImageDataGenerator(rescale = 1 / 255)
train_generator = train_datagen.flow_from_directory(
    '/Week3/horse-or-human',
    target_size = (300, 300),
    batch_size = 128,
    class_mode = 'binary'
)
```

```
Found 1027 images belonging to 2 classes.
```

[15]:

```
history = model.fit(
    train_generator,
    steps_per_epoch = 8,
    epochs = 25,
    verbose = 1
)
```

```
Epoch 1/25
8/8 [==============================] - 44s 6s/step - loss: 0.6479 - accuracy:
0.7063
Epoch 2/25
8/8 [==============================] - 43s 5s/step - loss: 0.6058 - accuracy:
0.7063
Epoch 3/25
8/8 [==============================] - 41s 5s/step - loss: 0.5449 - accuracy:
0.7820
Epoch 4/25
8/8 [==============================] - 41s 5s/step - loss: 0.7051 - accuracy:
0.8398
Epoch 5/25
8/8 [==============================] - 42s 5s/step - loss: 0.2895 - accuracy:
0.8676
Epoch 6/25
8/8 [==============================] - 52s 6s/step - loss: 0.1596 - accuracy:
0.9477
Epoch 7/25
8/8 [==============================] - 45s 6s/step - loss: 0.6036 - accuracy:
0.9132
Epoch 8/25
8/8 [==============================] - 51s 6s/step - loss: 0.2052 - accuracy:
0.9404
Epoch 9/25
8/8 [==============================] - 51s 6s/step - loss: 0.2642 - accuracy:
0.9266
```

```
Epoch 10/25
8/8 [==============================] - 45s 6s/step - loss: 0.1007 - accuracy:
0.9633
Epoch 11/25
8/8 [==============================] - 40s 5s/step - loss: 0.0520 - accuracy:
0.9822
Epoch 12/25
8/8 [==============================] - 40s 5s/step - loss: 0.8376 - accuracy:
0.8865
Epoch 13/25
8/8 [==============================] - 43s 5s/step - loss: 0.0886 - accuracy:
0.9655
Epoch 14/25
8/8 [==============================] - 48s 6s/step - loss: 0.0368 - accuracy:
0.9922
Epoch 15/25
8/8 [==============================] - 40s 5s/step - loss: 0.0285 - accuracy:
0.9911
Epoch 16/25
8/8 [==============================] - 41s 5s/step - loss: 0.0302 - accuracy:
0.9922
Epoch 17/25
8/8 [==============================] - 40s 5s/step - loss: 0.4362 - accuracy:
0.8765
Epoch 18/25
8/8 [==============================] - 43s 5s/step - loss: 0.0549 - accuracy:
0.9855
Epoch 19/25
8/8 [==============================] - 48s 6s/step - loss: 0.0109 - accuracy:
0.9980
Epoch 20/25
8/8 [==============================] - 40s 5s/step - loss: 0.0087 - accuracy:
0.9978
Epoch 21/25
8/8 [==============================] - 41s 5s/step - loss: 0.0019 - accuracy:
1.0000
Epoch 22/25
8/8 [==============================] - 41s 5s/step - loss: 7.2441e-04 -
accuracy: 1.0000
Epoch 23/25
8/8 [==============================] - 45s 6s/step - loss: 0.3931 - accuracy:
0.9366
Epoch 24/25
8/8 [==============================] - 56s 7s/step - loss: 0.2940 - accuracy:
0.9433
Epoch 25/25
8/8 [==============================] - 45s 6s/step - loss: 0.1725 - accuracy:
0.9544
```

[16]:

```
model.save("horse_or_human.h5")
print("Saved Model to Disk")
```

```
Saved Model to Disk
```

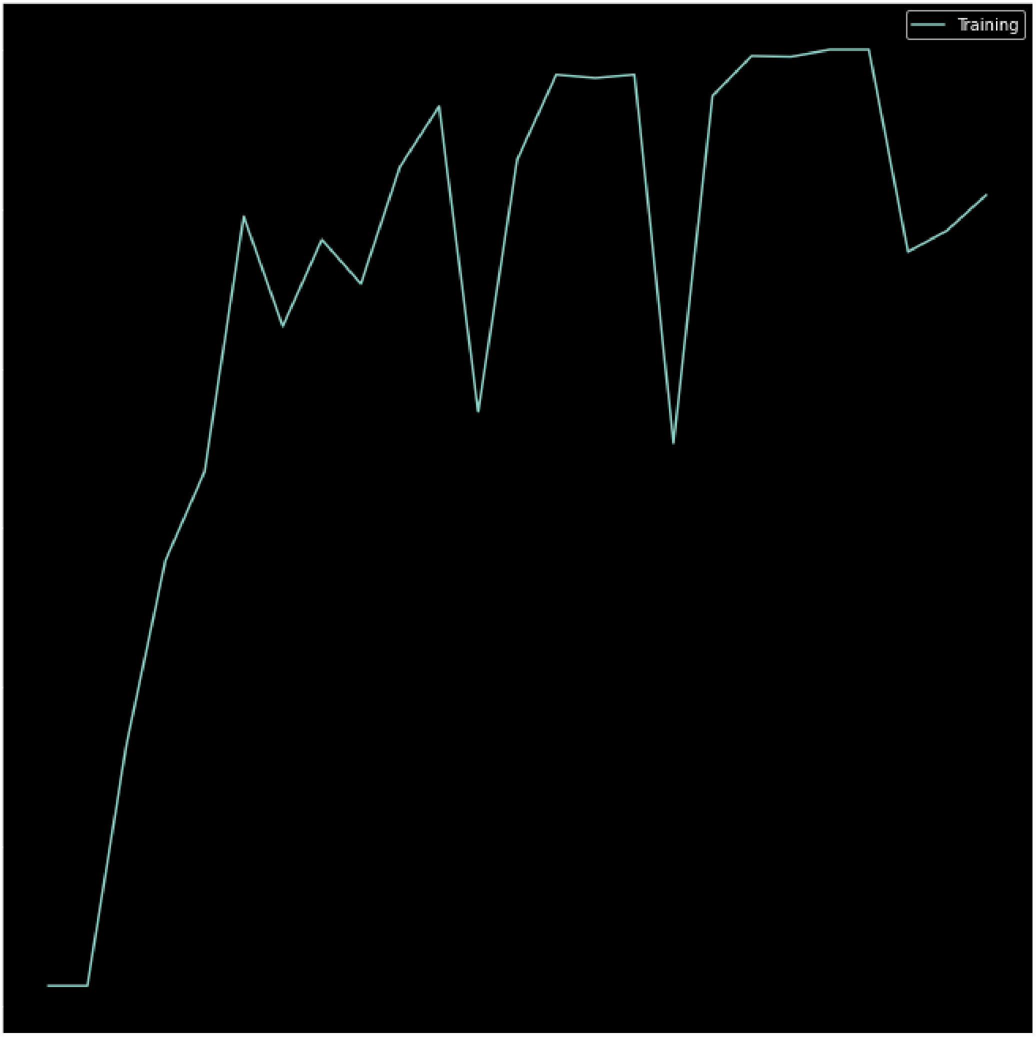
[26]:

```
# predicting images
path = 'F:/Courses/COURSERA/Tensorflow in practice/Course1/Week3/human.jpg'
img = image.load_img(path, target_size=(300, 300))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
images = np.vstack([x])
classes = model.predict(images, batch_size=10)
print(classes[0])
if classes[0]>0.5:
    print("It is a human")
else:
    print("It is a horse")
```

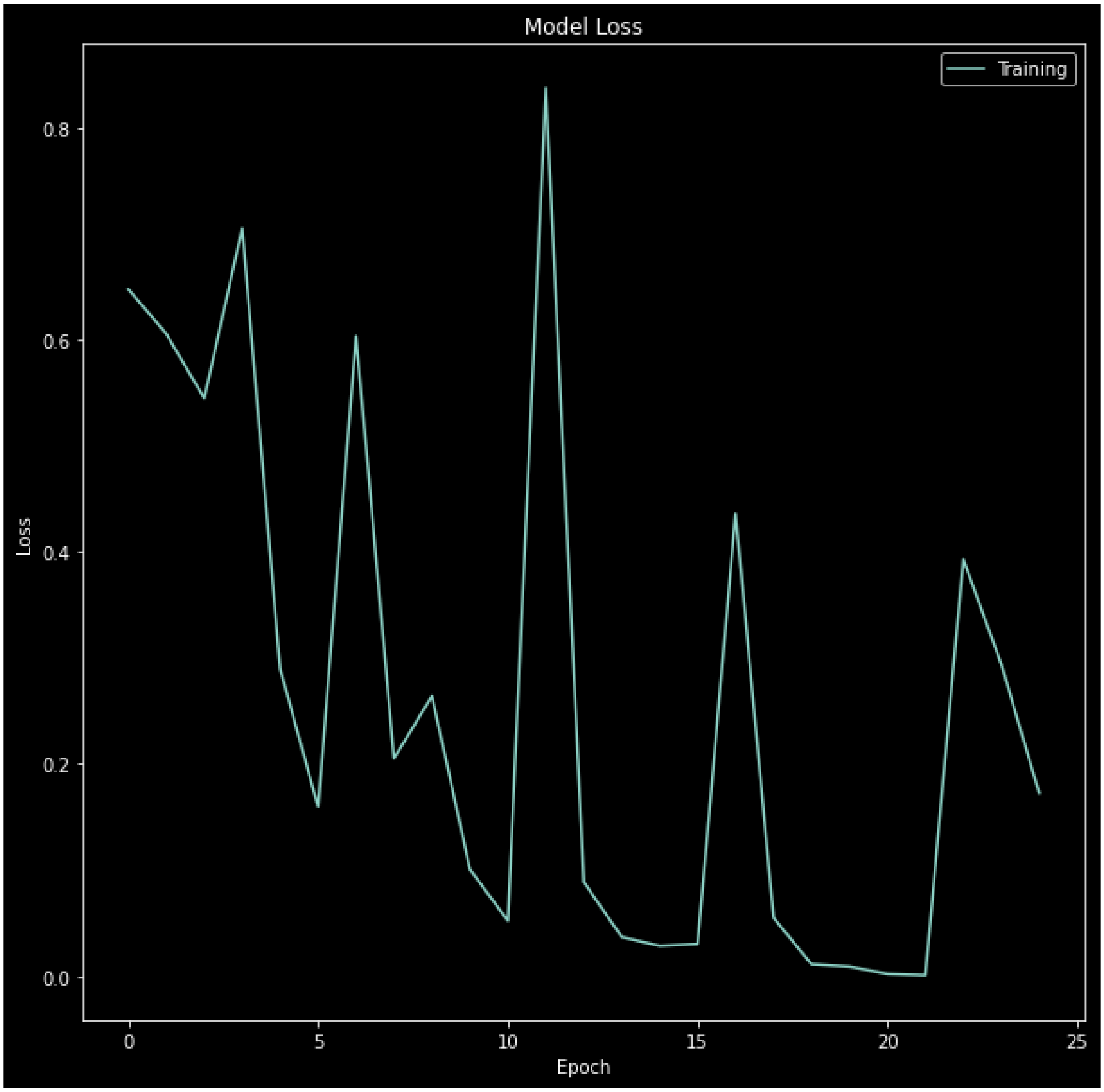
```
[1.]
It is a human
```

[27]:

```
plt.figure(figsize=(10,10))
plt.style.use('dark_background')
plt.plot(history.history['accuracy'])
# plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Training', 'Testing'])
plt.tight_layout()
plt.show()
```



[28]:
```
plt.figure(figsize=(10,10))
plt.style.use('dark_background')
plt.plot(history.history['loss'])
# plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Training', 'Testing'])
plt.show()
```


Model Loss
Training
Loss
0.0
0.2
0.4
0.6
0.8
Epoch
0
5
10
15
20
25

```
[ ]:
```